

REORDER ENGINE WITH ERROR RECOVERY

BACKGROUND OF THE INVENTION

A. Field of the Invention

[0001] The present invention relates generally to data processing and, more particularly, to reordering of information that arrives out of order after transmission.

B. Description of Related Art

[0002] Routers receive data on a physical media, such as optical fiber, analyze the data to determine its destination, and output the data on a physical media in accordance with the destination. Routers were initially designed using a general purpose processor executing large software programs. As line rates and traffic volume increased, however, general purpose processors could not scale to meet these new demands. For example, as functionality was added to the software, such as accounting and policing functionality, these routers suffered performance degradation. In some instances, the routers failed to handle traffic at line rate when the new functionality was turned on.

[0003] To meet the new demands, purpose-built routers were designed. Purpose-built routers are designed and built with components optimized for routing. They not only handled higher line rates and higher network traffic volume, they also added functionality without compromising line rate performance.

[0004] A purpose-built router may include a number of input and output ports from which it transmits and receives information packets. A switching fabric or other transmission medium may be implemented in the router to carry the packets between the ports. In a high-performance purpose-built router, the switching fabric may transmit a large amount of information between a number of internal components. Typically, the information is transmitted in discrete quantities called packets, or broken down even further into a series of cells.

[0005] One property of the switching fabric is that, even though it may have enough bandwidth to handle its cell traffic, cells transmitted to a particular destination may arrive at that destination out of order. Thus, although a first cell may be transmitted before a second cell, the second cell may be received before the first cell. Generally, the order of the received cells is important and should be preserved.

[0006] Reorder circuits may be used to reorder cells received from a particular source into their correct transmission order. Essentially, each cell, as it is transmitted from a source, is appended with a sequence number. The reorder circuits examine the received sequence numbers to determine the correct cell order.

[0007] One implementation of a reorder circuit is as a circular buffer in which arriving cells are placed at positions in the buffer based on the cell's sequence number. A pointer register keeps track of the rear-most position in the buffer. When a cell arrives that corresponds to the position in the pointer register, the reorder circuit increments the pointer value to point to the next open

position in the buffer and processes all cells between the old pointer position and the new pointer position.

[0008] Typically, out-of-order cells are received in a relatively close sequence to their true order. Occasionally, however, a cell may arrive that is significantly out-of-order (e.g., the cell is delayed much more than normal). Designing a reorder circuit to be able to reorder such pathological worst case conditions can significantly increase the memory size requirements of the reorder circuit.

[0009] Thus, there is a need in the art for reorder circuits to be able to effectively handle worst case out-of-order scenarios.

SUMMARY OF THE INVENTION

[0010] Structure and processes consistent with the principles of the invention, among other things, efficiently reorder information streams that occasionally include highly out-of-order data items.

[0011] In one aspect consistent with the principles of the invention, a reordering device includes a reorder buffer, a reorder buffer pointer, and a reorder engine. The reorder buffer stores information relating to data items, each of the data items including a sequence number chosen from a sequence number space and that indicates a relative order of the data items. The reorder buffer pointer indicates a rearmost active entry of the reorder buffer relative to the sequence number space. The reorder engine classifies data items based on the sequence number of the data item and the value of the reorder buffer pointer.

Further, the reorder engine determines whether to store the data items in the reorder buffer based on the classification.

[0012] A second aspect of the invention is directed to a method of ordering data items. The method includes receiving the data items, where the received data items are associated with a sequence number chosen from a sequence number space. The sequence number indicates a relative order of the data items. The method classifies the received data items as one of a number of possible classifications based on the sequence number of the data items and a value that defines a position of a reorder buffer in the sequence number space. The data items are processed based on the classifications.

[0013] Another aspect of the invention is directed to a network device including a data transmission component and processing elements connected by the data transmission component. The processing elements communicate with one another by transmitting data items over the data transmission component. The processing elements each include a reorder component configured to arrange received data items into an order corresponding to an order in which the data items were transmitted. Each of the reorder components includes a reorder buffer and a reorder engine. The reorder buffer stores the data items, each of which includes a sequence number chosen from a sequence number space. The sequence number indicates the order of the data items. Further, the reorder engine classifies the data items based on the sequence number of the data items and based on a position of the reorder buffer relative to the sequence number

space, and the reorder engine determines whether to store the data items in the reorder buffer based on the classification of the data item.

BRIEF DESCRIPTION OF THE DRAWINGS

[0014] The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate an embodiment of the invention and, together with the description, explain the invention. In the drawings,

[0015] Fig. 1 is a block diagram illustrating an exemplary routing system in which systems and methods consistent with the principles of the invention may be implemented;

[0016] Fig. 2 is a detailed block diagram illustrating portions of the routing system shown in Fig. 1;

[0017] Fig. 3 is a diagram illustrating portions of the communication component illustrated in Fig. 2 and relating to the reception and reordering of data cells;

[0018] Fig. 4 is a diagram that conceptually illustrates the relationship between the entries in a reorder buffer and sequence numbers associated with received cells;

[0019] Figs. 5A and 5B are diagrams conceptually illustrating regions to which the reorder engine classifies received cells;

[0020] Fig. 6 is a flow chart illustrating operation of the reorder engine in handling received cells; and

[0021] Fig. 7 is a diagram illustrating a reorder engine and a reorder buffer consistent with an alternate embodiment of the invention.

DETAILED DESCRIPTION

[0022] The following detailed description of the invention refers to the accompanying drawings. The same reference numbers may be used in different drawings to identify the same or similar elements. Also, the following detailed description does not limit the invention. Instead, the scope of the invention is defined by the appended claims and equivalents.

[0023] As described herein, a reorder engine places data items of a received stream into the order in which they were transmitted. The reorder engine uses a reorder buffer to store incoming data items in a buffer location determined by a sequence number contained in the data items. Incoming data units are classified by the reorder engine into one of three different regions depending on the data item's sequence number. Data items in the first region are inserted into the buffer. Data items in the second region are inserted into a shifted version of the buffer. Data items in the third region are considered non-recoverable errored data items and are dropped.

SYSTEM DESCRIPTION

[0024] Fig. 1 is a block diagram illustrating an exemplary routing system 42 in which the principles of the present invention may be implemented. System 42 includes packet forwarding engines (PFEs) 44, 46 ... 48, a switch fabric 50,

and a routing engine (RE) 52. System 42 receives a data stream from a physical link, processes the data stream to determine destination information, and transmits the data stream out on a link in accordance with the destination information.

[0025] RE 52 performs high-level management functions for system 42. For example, RE 52 communicates with other networks and systems connected to system 42 to exchange information regarding network topology. RE 52 creates routing tables based on network topology information, creates forwarding tables based on the routing tables, and forwards the routing tables to PFEs 44, 46, and 48. The PFEs use the routing tables to perform route lookups for incoming packets. RE 52 also performs other general control and monitoring functions for system 42.

[0026] PFEs 44, 46, and 48 are each connected to RE 52 and switch fabric 50. PFEs 44, 46, and 48 receive data at ports on physical links connected to a network, such as a wide area network (WAN). Each physical link could be one of many types of transport media, such as optical fiber or Ethernet cable. The data on the physical link is formatted according to one of several protocols, such as the synchronous optical network (SONET) standard, an asynchronous transfer mode (ATM) technology, or Ethernet.

[0027] PFE 44 will be used, by way of example, to discuss the operations performed by PFEs 44, 46, and 48 consistent with the principles of the invention. PFE 44 processes incoming data by stripping off the data link layer. PFE 44

converts header information from the remaining data into a data structure referred to as a notification.

[0028] For example, in one embodiment, the data remaining after the data link layer is stripped off is packet data. PFE 44 converts the layer 2 (L2) and layer 3 (L3) packet header information included with the packet data into a notification. PFE 44 stores the notification, some control information regarding the packet, and the packet data in a series of cells. In one embodiment, the notification and the control information are stored in the first two cells of the series of cells. The cells may be discrete instructions each having header and possibly trailer fields, or may simply be a bit stream processed in 64-byte segments.

[0029] PFE 44 performs a route lookup using the notification and the forwarding table from RE 52 to determine destination information. PFE 44 may also process the notification to perform protocol-specific functions, policing, and accounting, and might even modify the notification to form a new notification.

[0030] If the determined destination indicates that the packet should be sent out on a physical link connected to PFE 44, then PFE 44 retrieves the cells for the packet, converts the notification or new notification into header information, forms a packet using the packet data from the cells and the header information, and transmits the packet from the port associated with the physical link.

[0031] If the destination indicates that the packet should be sent to another PFE via switch fabric 50, then PFE 44 retrieves the cells for the packet, modifies

the first two cells with the new notification and new control information, if necessary, and sends the cells to the other PFE via switch fabric 50. Before transmitting the cells over switch fabric 50, PFE 44 appends a sequence number to each cell, which allows the receiving PFE to reconstruct the order of the transmitted cells. The receiving PFE reconstructs the cells in to their correct order and reconstructs the packet using the packet data from the cells, and sends the packet out on the port associated with the appropriate physical link of the receiving PFE.

[0032] In summary, RE 52, PFEs 44, 46, and 48, and switch fabric 50 perform routing based on packet-level processing. The PFEs store each packet using cells while performing a route lookup using a notification, which is based on packet header information. A packet might be received on one PFE and go back out to the network on the same PFE, or be sent through switch fabric 50 to be sent out to the network on a different PFE.

[0033] Fig. 2 is detailed a block diagram illustrating portions of routing system 42. PFEs 44, 46, and 48 connect to one another through switch fabric 50. Each of the PFEs may include one or more physical interface cards (PICs) 201-202 and flexible port concentrators (FPCs) 205.

[0034] PICs 201 and 202 may transmit data between a WAN physical link and FPC 205. Different PICs may be designed to handle different types of WAN physical links. For example, PIC 201 may be an interface for an optical link while PIC 202 may be an interface for an Ethernet link. Although Fig. 2 shows two

PICs connected to the FPCs, in other embodiments consistent with principles of the invention there can be more or fewer PICs connected to an FPC.

[0035] FPCs, such as FPC 205, handle packet transfers to and from PICs 201 and 202, and switch fabric 50. For each packet it handles, FPC 205 performs the previously discussed route lookup function. FPCs 205 communicate with switch fabric 50 through a fabric communication component 207 (labeled as Ncomp). Communication component 207 handles the protocols associated with transmitting cells to and receiving cells from switch fabric 50. In one embodiment, switch fabric 50 supports 144 communication components, implemented in 144 FPCs 205.

[0036] As previously mentioned, cells transmitted by a source communication component 207 over switch fabric 50 may be received out of order at the destination communication component. The destination communication component reorders the received cells into their appropriate order.

REORDER COMPONENT

[0037] Fig. 3 is a diagram illustrating portions of communication component 207 relating to the reception and reordering of data cells (or more generally, any unit of data) from switch fabric 50. Communication component 207 includes a memory interface 305 and a reorder component 310. Communication component 207 stores received cells in external memory 315. In one embodiment, memory 315 is a distributed memory.

[0038] Cells from switch fabric 50 are received at memory interface 305, which extracts basic cell identification information from the received cells. In one embodiment, each cell is labeled with a cell type (CT) identifier, such as "FC" (first cell of packet), "SC" (second cell of packet), "MC" (a middle cell of a packet), and "LC" (last cell of a packet). Memory interface 305 extracts this cell type identifier and the sequence number from the received cells, and stores the remaining portion of the cell in memory 315. Reorder component 310 receives the cell type identifier, the cell address in memory 315, and the sequence number from memory interface 305. Based on these three pieces of cell identification information, reorder component 310 determines the correct order for the incoming cells and outputs, in the correct cell order, each cell's address information.

[0039] Reorder component 310 includes a reorder pointer (Rptr) register 321 (Rptr), reorder engine 322, and a reorder buffer 323. Reorder engine 322 places cell information into reorder buffer 323 based on the cell sequence number and the value in Rptr 321.

[0040] In one embodiment, reorder buffer 323 is an array of 64 entries (labeled as entries 325). Each entry 325 stores information corresponding to a cell (or, more generally, any data unit) from reorder engine 322. One of entries 325 (entry zero) is shown in detail in Fig. 3, and, may store the cell address 331, cell type 332, and validity state 333 of the cell. Validity state 333 may be stored as a valid ("V") bit that indicates whether there is valid information in the

corresponding entry. A "1," for example, may indicate that the entry contains valid cell information while a "0" may indicate that the entry is empty.

[0041] Reorder buffer 323 is implemented as a circular buffer. Rptr 321 stores a value that points to the rearmost active entry in reorder buffer 323. New cells are written to reorder buffer 323 within an active window of entries that begins at the entry referenced by Rptr 321 and extends through a predetermined number of entries (e.g., all 64 entries). Reorder engine 322 determines the location to write the new cells in reorder buffer 323 based on the cell sequence number. When a cell is received which has a sequence number that corresponds to the value of Rptr (i.e., the cell belongs at the rearmost active entry in reorder buffer 323), reorder engine 322 processes the received cell as the next cell in the correct order of received cells. The processing may include outputting cell address 331 and cell type 322, and clearing cell validity bit 333. Reorder engine 322 may then increment Rptr. Because reorder buffer 323 is a circular buffer, Rptr 321 wraps around to the first entry in the reorder buffer 323 after reaching the last entry in the reorder buffer 323.

[0042] Fig. 4 is a diagram that conceptually illustrates the relationship between the entries in reorder buffer 323 and the sequence numbers associated with received cells. As previously mentioned, transmitting devices append a sequence number to cells transmitted through switch fabric 50. The range of possible sequence numbers, called the sequence number space, is larger than the size of reorder buffer 323. In the embodiment illustrated in Fig. 4, the sequence number space is 2048 possible numbers (the complete circle 400).

Thus, a transmitting device may append the sequence number zero to its first transmitted cell, sequence number one to the second transmitted cell, etc., up through sequence number 2047. The next appended sequence number will again be zero.

[0043] The active window defined by reorder array 323 is illustrated by slice 401 of circle 400. As Rptr is incremented, slice 401 moves clockwise around the circle.

[0044] Because the number of available sequence numbers can be much greater than the number of entries 325 in reorder buffer 323 (e.g., 64 entries in the reorder buffer and 2048 possible numbers in the sequence space), the appropriate address in the reorder buffer 323 (e.g., zero to 63) to store a cell is based on a modulo operation. Specifically, reorder engine 322 calculates the physical address in buffer 323 based on the sequence number modulo the number of entries in the buffer memory. Thus, in the example given above, the reorder engine calculates the address in reorder buffer 323 as the sequence number module 64.

[0045] In normal operation, the sequence number of incoming cells falls within slice 401. Whenever a cell is received at Rptr 321, reorder engine 322 processes that cell as the next in-sequence cell and increments Rptr, which effectively slides slice 401 one position clockwise in sequence space 400. Occasionally, however, a sequence number will arrive that is outside of slice 401. In this situation, reorder engine 322 handles the cell based on where in the sequence number space, relative to slice 401, the cell arrives. More specifically,

consistent with principles of the invention, reorder engine 322 classifies each incoming cell as belonging to one of three regions.

[0046] Figs. 5A and 5B are diagrams conceptually illustrating the three regions to which reorder engine 322 classifies each cell. The first region, labeled as region 501, maps to the physical reorder buffer 323. The second region, labeled as region 502, lies ahead (in the sequence number space) of the first region. The third region, labeled as region 503, lies just behind the first region. Stated differently, the second region extends for a predetermined count of sequence numbers (e.g., 100) after first region 501, while the third region encompasses the rest of the sequence space.

[0047] In general, reorder engine 322 classifies incoming cells as belonging to one of the three regions 501, 502, or 503, based on the sequence number associated with the cells, and handles the received cells differently based on the classification.

[0048] Fig. 6 is a flow chart illustrating operation of the reorder engine 322 in handling received cells based on their region classification.

[0049] For each received cell, reorder engine 322 examines the cell's sequence number and determines if the cell is in first region 501, second region 502, or third region 503 (act 601). Generally, the cell will belong to the first region. In this situation, reorder engine 322 stores the cell information in the appropriate entry in reorder buffer 323. More specifically, reorder engine 322 sets the valid bit corresponding to the cell and stores the cell address and cell type information in reorder buffer 323 (act 602). If the valid bit is set for the cell

corresponding to Rptr, reorder engine 322 processes that cell by outputting the cell address for the cell (acts 603 and 604). Reorder engine 322 then advances Rptr 321 and resets the valid bit corresponding to the processed cell (acts 605 and 606).

[0050] A cell arriving in the second region has arrived too early with respect to its logically neighboring cells. In this situation, reorder engine 322 advances Rptr just enough so that the last entry in reorder buffer 323 accommodates the new cell (acts 601 and 607). Alternatively, reorder engine 322 advances Rptr just enough so that the last entry in reorder buffer 323 accommodates the new cell and the cell referenced by Rptr is of cell type FC (first cell). In either case, this effectively shifts the window defined by the reorder buffer in the sequence space. This shifting is illustrated graphically in Fig. 5B.

[0051] Referring to Fig. 5B, assume that a new cell arrives with a sequence number that places it at position "x" 510. Reorder engine 522 shifts Rptr in the global sequence number space so that the new region just reaches newly arriving cell 510. Cells in the reorder engine that were skipped (in slice 512) are invalidated (act 608). The invalidation can be performed by clearing valid bits 333 of the invalidated cells. At this point, reorder engine 322 continues operating as if the new cell had arrived in the first region and perform Acts 602-606.

[0052] A cell arriving in the third region has arrived too late and is ignored by the reorder engine (act 609).

[0053] In the event of a flush of shifted region 512 (act 608), reorder engine 322 invalidates valid bits 333 corresponding to each flushed cell. In some implementations, however, discarding a received cell may require more than simply resetting a valid bit. For example, the system may need to signal memory 315 that the memory allocated to the discarded cell can be reclaimed. In this situation, due to the latency associated with deallocating memory, it may be infeasible to perform Act 608 for a large number of flushed cells without delaying later portions of the system.

[0054] Fig. 7 is a diagram illustrating a reorder engine 722 and a reorder buffer 723 for discarding multiple cells for situations in which it may not be feasible to simultaneously discard multiple cells. In a manner similar to reorder buffer 323, entries in reorder buffer 723 are associated with a cell address 731, a cell type field 732, and a valid field 733. Additionally, each entry is associated with a flush ("F") bit 734. In operation, reorder engine 722 may perform the acts described above. However, when invalidating a cell, reorder engine 722, in addition to resetting valid field 733 corresponding to the cell, sets flush bit 734 that corresponds to the cell. A set flush bit indicates that additional processing, such as reclaiming memory associated with the cell, is still to be performed. Reorder engine 722 may perform this additional processing at any convenient time. For example, when a new cell arrives and is placed in reorder buffer 723, reorder engine 722 may clear the flush bit and reclaim the memory associated with the previously invalidated cell. Because only one cell can arrive at any particular time, this technique automatically serializes memory recovery.

SUMMARY

[0055] By classifying incoming cells as belonging to either the first, second, or third region, reorder engine 322 can effectively handle error conditions. In normal operation, the vast majority of incoming cells will fall in the first region and can thus be processed normally. Cells arriving in the second region indicate that a previous cell is late or delayed. If this previous cell is delayed but does eventually arrive, it will arrive in the third region and is simply ignored.

[0056] Although described in the context of a purpose-built router, concepts consistent with the principles of the invention can be implemented in any system that requires high performance data item processing. Apparatus, systems, and methods based on the principles of the pipeline engine described herein may be used in any environment for sequencing data items associated with an entity.

[0057] The foregoing description of preferred embodiments of the present invention provides illustration and description, but is not intended to be exhaustive or to limit the invention to the precise form disclosed. Modifications and variations are possible in light of the above teachings or may be acquired from practice of the invention. For example, although memory 315 was described as being implemented externally from reorder component 310, memory 315 may be integrated with reorder component 310.

[0058] No element, act, or instruction used in the description of the present application should be construed as critical or essential to the invention unless explicitly described as such. Also, as used herein, the article "a" is intended to include one or more items. Where only one item is intended, the term "one" or similar language is used. The scope of the invention is defined by the claims and their equivalents.